For More Information

Louise E. Moser

Computer Networks and Distributed Systems Laboratory
Department of Electrical and Computer Engineering
University of California, Santa Barbara 93106

(805) 893-4897

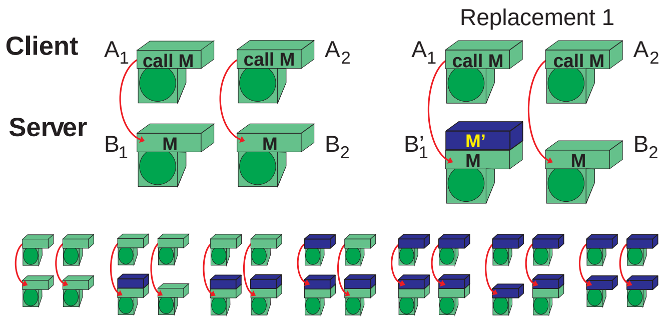
moser@ece.ucsb.edu

http://www.ece.ucsb.edu/Faculty/moser/default.html

How Far Have We Come?

- **Developing complex application software is hard**
- **Using Totem, Eternal can make it easier by**
  - **Hiding distribution, replication, consistency, and the handling of faults**
  - **Exploiting replication to provide fault tolerance and to allow the system to be evolved dynamically**
  - **Providing location transparency and interoperability using commercial CORBA ORBs**
- **Eternal incurs reasonable overheads for interception, replication and multicasting**
- **Eternal will enable us to build fault-tolerant, evolvable distributed application systems that can run forever**

Evolution of Objects and Interfaces



Evolution of Objects and Interfaces

To add a parameter to an existing method

- Create a new method with the additional parameter
- Replace server replicas, one at a time, with replicas that use both old and new methods
- Replace client replicas, one at a time, with new replicas that invoke the new method
- Replace server replicas, one at a time, with replicas that use only the new method

The Eternal Evolution Manager

- Exploits replication to allow both hardware and software to be removed and replaced, while the system is live
- CORBA’s interoperability permits new hardware to be substantially different from old hardware
- Upgrades to the software objects are performed, one replica at a time, with state transfers from existing replicas to new replicas
- Upgrades to the software objects and interfaces require a sequence of replacements
- The Eternal Evolution Manager is programmed, and executes, as CORBA objects

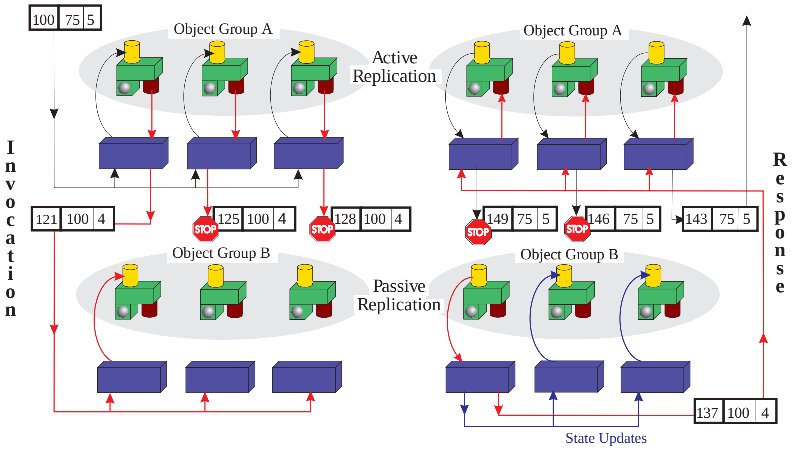
The Eternal Resource Manager

- Distributes replicated objects across the system to meet fault-tolerance and load-balancing objectives
- Specifications in the Interface Repository
  - Quality-of-service requirements, such as real-time response time
  - Resources needed by each object for each operation
  - Capabilities of available processor and network resources
- The Eternal Resource Manager is programmed, and executes, as CORBA objects

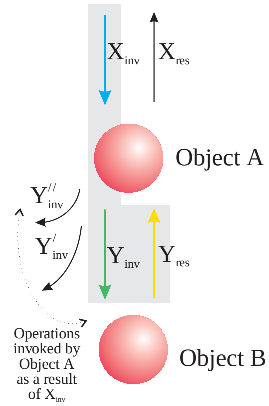
Exception Handling in Eternal

- Application-specific exceptions
  - Exactly the same for all replicas
  - Returned to the client object as the result of its invocation
- Platform-specific exceptions
  - May be different for different platforms
  - Handled locally and not returned to client object
  - May result in the local replica of the server object being terminated
  - May involve a recovery action, including the creation of a new replica by Eternal

Use of Operation Identifiers



Operation Identifiers



Invocation Identifier for Operation Y

Timestamp assigned by Totem to message Y_{inv}	Timestamp assigned by Totem to message X_{inv}	Sequence number assigned by Eternal to operation Y

Unique Operation Identifier

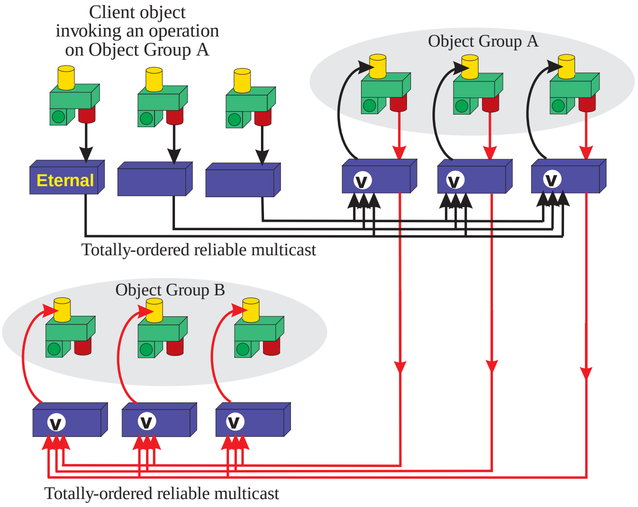
Timestamp assigned by Totem to message Y_{res}	Timestamp assigned by Totem to message X_{inv}	Sequence number assigned by Eternal to operation Y

Response Identifier for Operation Y

Need for Operation Identifiers

- In passive replication, to detect and suppress duplicates when
  - Primary replica invokes an operation and then fails, leading to possible reinvocation after the election of a new primary
- In active replication, to detect and suppress duplicates when
  - Multiple replicas invoke the same operation
  - Multiple replicas respond with the same results
- In majority voting, to ensure that
  - Voting algorithm votes on the same invocations and the same responses

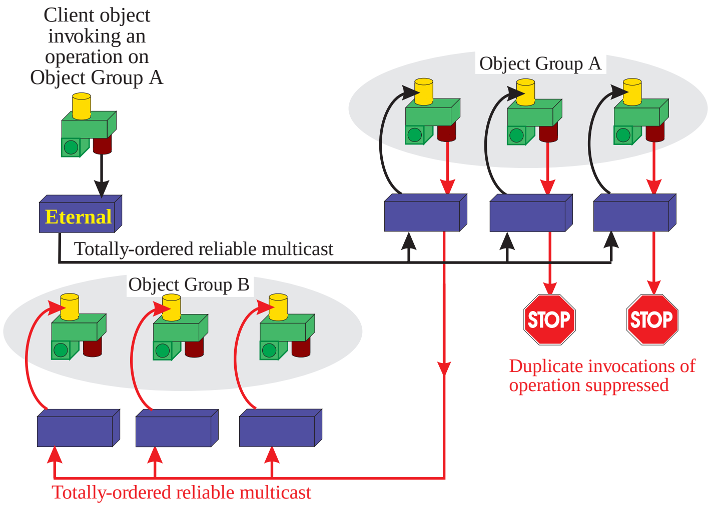
Majority Voting



Majority Voting

- Active replication with majority voting on each invocation and response
- Application is shielded from crash, timing, omission and commission faults
- Needs a more robust group communication protocol that also protects against commission faults
- Computational cost – cost of performing the operation by each replica plus the cost of majority voting by the Eternal Replication Manager
- Communication cost
  - Three or more multicast messages for invocation
  - Three or more multicast messages for response

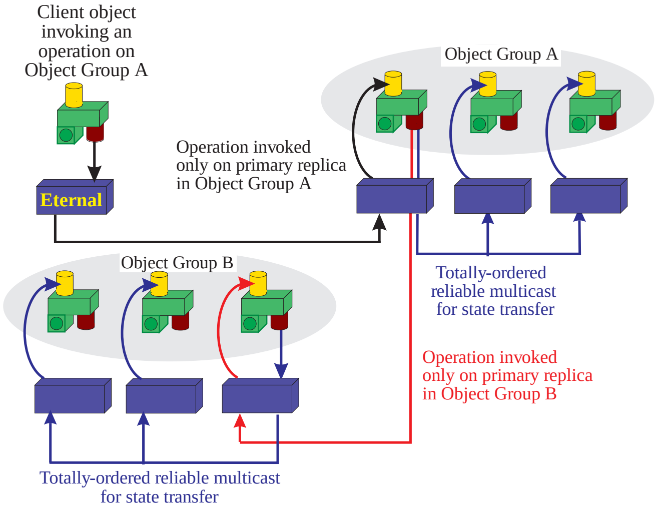
Active Replication



Active Replication

- Each replica of an object performs the same operations in the same order, thereby guaranteeing consistency of the states of the replicas
- Failure of a server replica is transparent to a client object as long as there are two or more server replicas
- Application is shielded from crash, timing and omission faults
- Computational cost – each replica performs the operation
- Communication cost
  - One or more multicast messages for invocation
  - One or more multicast messages for response

Passive Replication



Client object invoking an operation on Object Group A

Operation invoked only on primary replica in Object Group A

Object Group A

Totally-ordered reliable multicast for state transfer

Object Group B

Operation invoked only on primary replica in Object Group B

Totally-ordered reliable multicast for state transfer

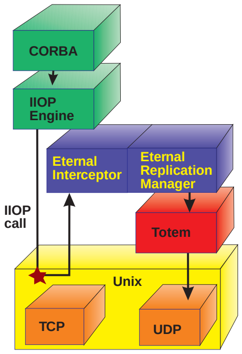
Passive Replication

- Single primary replica performs invoked operation and returns the results of the operation
- Eternal transfers state from the primary replica to the non-primary replicas at the end of the operation, thereby ensuring consistency
- Application is shielded from crash faults and timing faults
- Computational cost – only the primary performs the operation
- Communication cost
  - One multicast message for invocation
  - One multicast message for response
  - One or more multicast messages for state transfer, which may be significant if the state is large

The Eternal Replication Manager

- **Object group abstraction provides replication and group transparency**
- **Operations to be performed on an object are multicast to the object group (consisting of its replicas)**
- **Both client and server objects can be replicated**
- **Passive replication, active replication and majority voting are supported and can coexist**
- **Operation identifiers allow duplicate operations to be detected and suppressed**
- **Nested operations of any depth are supported**

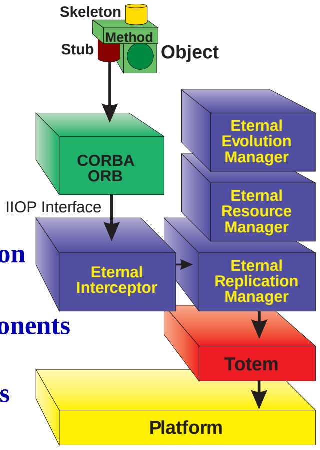
The Eternal Interceptor



- Commercial CORBA ORBs are complex and proprietary, but all provide IIOP over TCP
- But TCP does not provide total ordering of messages within multicast groups
- Intercepts the IIOP calls to TCP using the /proc facility of Unix
- Redirects the calls to Totem via the Eternal Replication Manager
- Works with Iona’s Orbix, Sun’s NEO, Visigenic’s VisiBroker, Expersoft’s CORBAPlus, HP’s ORBPlus, Chorus’ COOL, OOC’s OmniBroker, Xerox’s ILU, Olivetti’s Omni-ORB 2

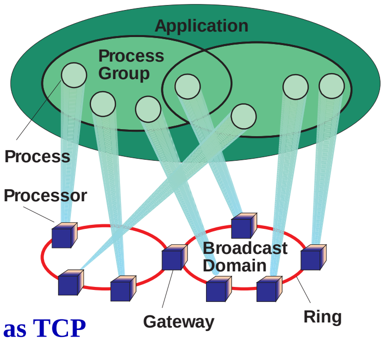
The Eternal System

- Implemented on top of Totem
- Application programmer interface is object-oriented, rather than message-oriented
- The Eternal system provides
  - Object replication and distribution
  - Fault detection and recovery
  - Continued operation in all components of a partitioned network
  - Hardware and software upgrades in a live system
- Uses standard IIOP interface, works with any commercial CORBA ORB



The Totem System

- Provides reliable totally ordered delivery of multicast messages to processes in process groups over single or multiple local-area networks
- Delivers messages in a consistent total order that respects causality despite
  - Message loss
  - Processor and process faults
  - Network partitioning
- Maintains the membership and topology of the network
- Delivers multicast messages as fast as TCP can deliver messages point-to-point



Maintaining Consistency with Multicast Messages

- Operations to be performed on the replicated objects are contained in messages that are multicast to the processors hosting the replicas, using a group communication system such as Totem
- All of the processors receive the multicast messages in the same total order
- The processors perform the same operations on the replicas in the same total order
- The Eternal system maintains consistency of the replicas
- The replicas of an object are presented as a single object to the application programmer

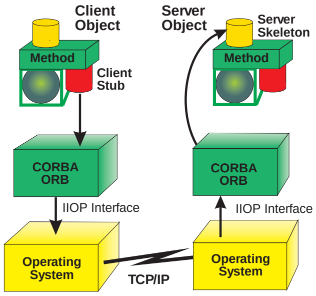
Simplifying the Application Programming

- Our objective is to build systems that are fault-tolerant, adaptive, and evolvable
  - This requires replication of processing and data
  - But, maintaining consistency of replicated data is difficult in the presence of asynchrony and faults
- Application programmers should not be concerned with these difficult system programming issues
- Our objective is to simplify the application programming by
  - Hiding the difficult issues of replication, consistency, fault detection and recovery
  - Separating the functional behavior of the program from the allocation and management of resources

Distributed Object Computing and CORBA

Common Object Request Broker Architecture (CORBA)

- Client requests a server to perform operations for it
- Requests are handled by the Object Request Broker (ORB) to achieve
  - Location transparency
  - Interoperability
- Internet Inter-Orb Protocol (IIOP) provides a clean simple interface for the invocation of operations



Where Are We Going?

- In the past, processing speed and memory capacity have been the limiting factors in computer systems
- In the present, with distributed networked computing, the limitation is network bandwidth
- But processors and networks are becoming faster, and memory is becoming cheaper
- In the future, the limiting factor will be the complexity of the application software

The Eternal System

Louise E. Moser

Department of Electrical and Computer Engineering
University of California, Santa Barbara

moser@ece.ucsb.edu
http://www.ece.ucsb.edu/Faculty/moser/default.html

Sponsor: Bob Laddaga DARPA ITO